A Quick Start Guide to Survey Research

Liz Carey (and hopefully many others)

2019-01-26

Contents

Welcome to survey research **5**

Preface **7**
Outline 7
Prerequisites 7
Acknowledgements 7

1 Designing a survey **9**
1.1 What is your research goal? 9
1.2 Who are you studying? 11

2 Writing effective survey questions **13**

3 Survey Analysis **15**
3.1 Organize your workspace 15
3.2 Data Cleaning 16

A Setting up R **25**
A.1 Package installation 25

B Setting up python **27**

C Generating fake data **29**

Welcome to survey research

This book is intended to be a quick resource for conducting survey research. By no means is it intended to be comprehensive of all survey research methodologies.

Preface

Hopefully you'll find this book to be a condensed and easy to read resource on survey research.

We developed this book in the hopes of future collaboration among other UX researchers.

Outline

The content of the book will include:

- **Chapter 1**
- **Chapter 2**

Prerequisites

All you need is an interest in conducting survey research and basic data analysis, we'll include code snippets (python and R) along the way.

Acknowledgements

This book wouldn't be possible without the contributions of:

Chapter 1

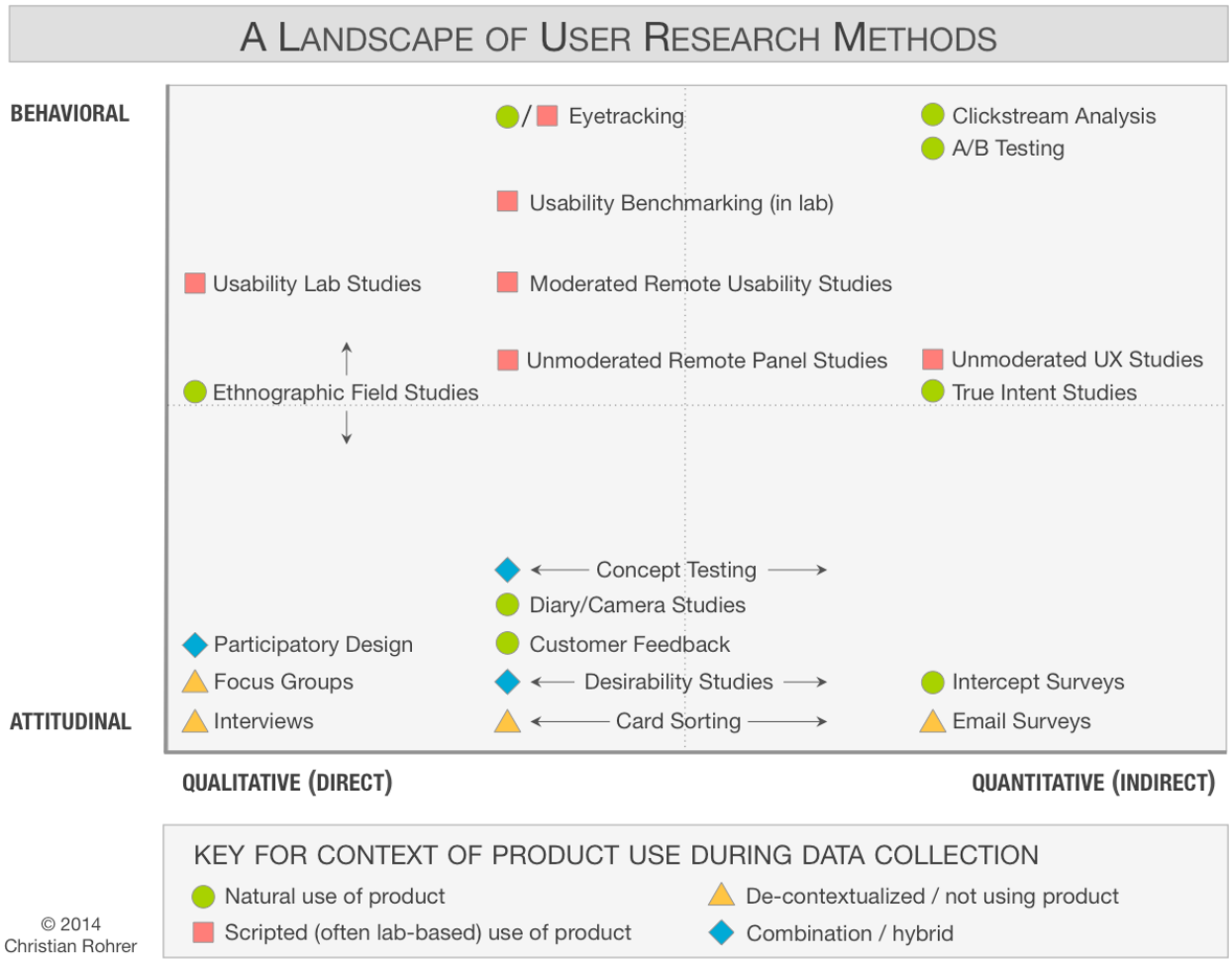
Designing a survey

1.1 What is your research goal?

First, establish if a survey is the right method to accomplish your research goal by asking yourself:

- What do you currently know?
- What *don't* you know?

Below is a useful visualization from the Nielsen Norman group on how to decide between which qualitative or quantitative methods to answer your research goal (Rohrer, 2014).



Surveys are great for answering the "How many and how much" of what people do and say; surveys are not the best method at understanding the "Why and how to fix" a product problem.

1.2 Who are you studying?

This question may be simple at first, but when you start to narrow down

Chapter 2

Writing effective survey questions

Effective survey questions result in **consistent** and **reliable** responses.

Surveys are NOT a shortcut for usability tests

Chapter 3

Survey Analysis

After you've fielded your survey, here are the steps to making sense of the data.

This section assumes you have a laptop set up to work with in either R or python. Head over to the Appendix page if you need help with set up.

3.1 Organize your workspace

Before beginning any analysis, you'll want to set up a reproducible workflow. Below is an adapted suggestion on how to organize your workspace from Ben Marwick, Carl Boettiger, and Lincoln Mullen (Ben Marwick, 2018). Keeping your workspace organized is the best way for you and others to understand and reproduce your analysis.

```
project
|- DESCRIPTION          # project metadata and dependencies
|- README.md            # top-level description of content and guide to users
|
|- data/                # data files used
|  +- raw_data.csv      # data files in open formats such as TXT, CSV, TSV, etc.
|  +- cleaned_data.csv  # data files that have been cleaned, merged, etc that you'll use for survey anal
|
|- analysis/            # any programmatic code
|  +- my_report.Rmd     # R markdown file with narrative text interwoven with code chunks
|  +- makefile          # builds a PDF/HTML/DOCX file from the Rmd, code, and data files
|  +- scripts/          # code files (R, shell, etc.) used for data cleaning, analysis and visualisatio
|  +- figures/          # saved outputs of your figures
|
|- R/
|  +- my_functions.R    # custom R functions that are used more than once throughout the project
|
|- man/
|  +- my_functions.Rd   # documentation for the R functions (auto-generated when using devtools)
|
```

R version

```r
#List the directory names you want to create
folder_names <- c("data",
                  "data/raw",
```

```
                    "data/clean",
                  "analysis",
                    "analysis/scripts",
                     "analysis/figures",
                  "R")

#Create the directories
sapply(folder_names, dir.create)
```

3.2 Data Cleaning

Before you can begin looking at the results, you'll need to clean the data. By "cleaning" the data, we mean edited the raw file into a format that will make the analysis valid and easier.

3.2.1 Load the data

Download your raw survey data as a csv and load it into your your analysis tool of choice (e.g. Ipython notebook or Rstudio)

R version

```
# load necessary packages for analysis
library(tidyverse)        #contains all the library packages to manipulate and transform data
library(summarytools)     #shortcut tools to visualize summaries of the data

# read/store the data as the variable df (short for dataframe)
# replace "file" with "https://raw.githubusercontent.com/lizmcarey/survey-guide/master/sample_data/Surv
file <- "./sample_data/Survey_test_data.csv" #load file from folder heirarchy
df <- read_csv(file)
```

python version

```
#load necessary modules for analysis
import pandas as pd

#read/store the data as the variable df (short for dataframe)
df = pd.read_csv(filename)
```

3.2.2 Loading Qualtrics data

When you download a csv from Qualtrics, it will come with a few extra rows you don't need. Here are some automated scripts you can add to your makefile to speed up your workflow

R version manual

```
# Store the column names by reading in the column header
df_names <- read_csv(file, n_max=0) %>% names()

# Read the entire file
df <- read_csv(file,
               col_names = df_names, # use df_names to title the columns
               skip = 3)             # skip the first three lines
```

```
#store the question names
question_bank <- read_csv(file, n_max=1) %>%  # read in the first row of the file
                 select(starts_with("Q")) %>% # select columns that start with Q
                 gather(key, question_text)   # transform data from wide to long
```

R version programmatic

```
#function to load qualtrics csv and remove extra rows
load_qualtrics_csv <- function(file) {
  df_names <- read_csv(file, n_max = 0) %>% names()

  df <- read_csv(file, col_names = df_names, skip = 3)
}

#function to store questions
get_questions <- function(file) {
  qb <- read_csv(file, n_max = 1) %>%
        select(starts_with("Q")) %>%
        gather(key, question_text)
}

#Use function to read in survey file, and skip first 3 lines
df <- load_qualtrics_csv(file)

#Use function to store question wording
question_bank <- get_questions(file)
```

3.2.3 Preview the data

It's important to get a look at the data to spot any errors in uploading the dataset and the validity of the responses.

You'll want to check for:

- Total number of observations/rows
- Duplicate responses
- Drop off/Abandon rate of the survey
- Average survey completion time
- "Speeders:" those who couldn't have completed the survey in a reasonable amount of time

---

There are multiple different ways to preview your dataset before analysis. One quick way is to check the first few rows of your data. You can do this with the function `head()`.

```
#Check the first 5 rows of data
head(df)
```

```
## # A tibble: 6 x 29
##   StartDate           EndDate             Status IPAddress Progress
##   <dttm>              <dttm>              <chr>  <lgl>        <dbl>
## 1 2019-01-15 13:28:39 2019-01-15 13:28:39 Surve~ NA             100
## 2 2019-01-15 13:28:40 2019-01-15 13:28:40 Surve~ NA             100
## 3 2019-01-15 13:36:47 2019-01-15 13:36:47 Surve~ NA             100
## 4 2019-01-15 13:36:47 2019-01-15 13:36:47 Surve~ NA             100
## 5 2019-01-15 13:36:48 2019-01-15 13:36:48 Surve~ NA             100
```

```
## 6 2019-01-15 13:36:48 2019-01-15 13:36:48 Surve~ NA             100
## # ... with 24 more variables: `Duration (in seconds)` <dbl>,
## #   Finished <lgl>, RecordedDate <dttm>, ResponseId <chr>,
## #   RecipientLastName <lgl>, RecipientFirstName <lgl>,
## #   RecipientEmail <lgl>, ExternalReference <lgl>, LocationLatitude <dbl>,
## #   LocationLongitude <dbl>, DistributionChannel <chr>,
## #   UserLanguage <lgl>, Q1 <chr>, Q2 <chr>, Q3_4 <chr>, Q3_5 <chr>,
## #   Q3_6 <chr>, Q3_7 <chr>, Q3_8 <chr>, Q3_9 <chr>, Q3_10 <chr>,
## #   Q3_10_TEXT <chr>, Q4 <chr>, Q5 <chr>
```

A more comprehensive way to view your dataset is with the `skimr` package. This package will give an overview of the number of observations and variables in your data.

The missing column should not be greater than 20% of your total number of observations (unless it's a multiselect question).

Questions with dropoff greater than 20% can signal that the question was difficult for respondents to answer; you should be wary of response bias and consider removing the question from analysis and rewording the question for future survey sends.

```
library(skimr)
skim(df)
```

```
## Skim summary statistics
##  n obs: 502
##  n variables: 29
##
## -- Variable type:character -----------------------------------------------
##             variable missing complete   n min max empty n_unique
##  DistributionChannel       0      502 502   4   4     0        1
##                   Q1       0      502 502   5  14     0        6
##                   Q2       0      502 502  18  34     0        5
##               Q3_10     184      318 502   5   5     0        1
##           Q3_10_TEXT     184      318 502  51 135     0      318
##                Q3_4     201      301 502  26  26     0        1
##                Q3_5     165      337 502  22  22     0        1
##                Q3_6     174      328 502  21  21     0        1
##                Q3_7     172      330 502  19  19     0        1
##                Q3_8     184      318 502  18  18     0        1
##                Q3_9     162      340 502  23  23     0        1
##                  Q4       0      502 502  11  22     0        7
##                  Q5       0      502 502  53 134     0      502
##          ResponseId       0      502 502  17  17     0      502
##              Status       0      502 502  11  11     0        1
##
## -- Variable type:logical -------------------------------------------------
##            variable missing complete   n mean           count
##   ExternalReference     502        0 502  NaN             502
##            Finished       0      502 502    1 TRU: 502, NA: 0
##           IPAddress     502        0 502  NaN             502
##      RecipientEmail     502        0 502  NaN             502
##  RecipientFirstName     502        0 502  NaN             502
##   RecipientLastName     502        0 502  NaN             502
##        UserLanguage     502        0 502  NaN             502
##
```

```
## -- Variable type:numeric -------------------------------------------------
##               variable missing complete   n     mean   sd      p0     p25
##  Duration (in seconds)       0      502 502    0.024 0.15    0       0
##       LocationLatitude       0      502 502   37.77  0      37.77   37.77
##      LocationLongitude       0      502 502 -122.41  0    -122.41 -122.41
##               Progress       0      502 502  100     0     100     100
##      p50     p75    p100     hist
##     0       0       1
##    37.77   37.77   37.77
##  -122.41 -122.41 -122.41
##   100     100     100
##
## -- Variable type:POSIXct -------------------------------------------------
##      variable missing complete   n        min        max     median
##       EndDate       0      502 502 2019-01-15 2019-01-15 2019-01-15
##  RecordedDate       0      502 502 2019-01-15 2019-01-15 2019-01-15
##     StartDate       0      502 502 2019-01-15 2019-01-15 2019-01-15
##  n_unique
##        74
##        74
##        74
```

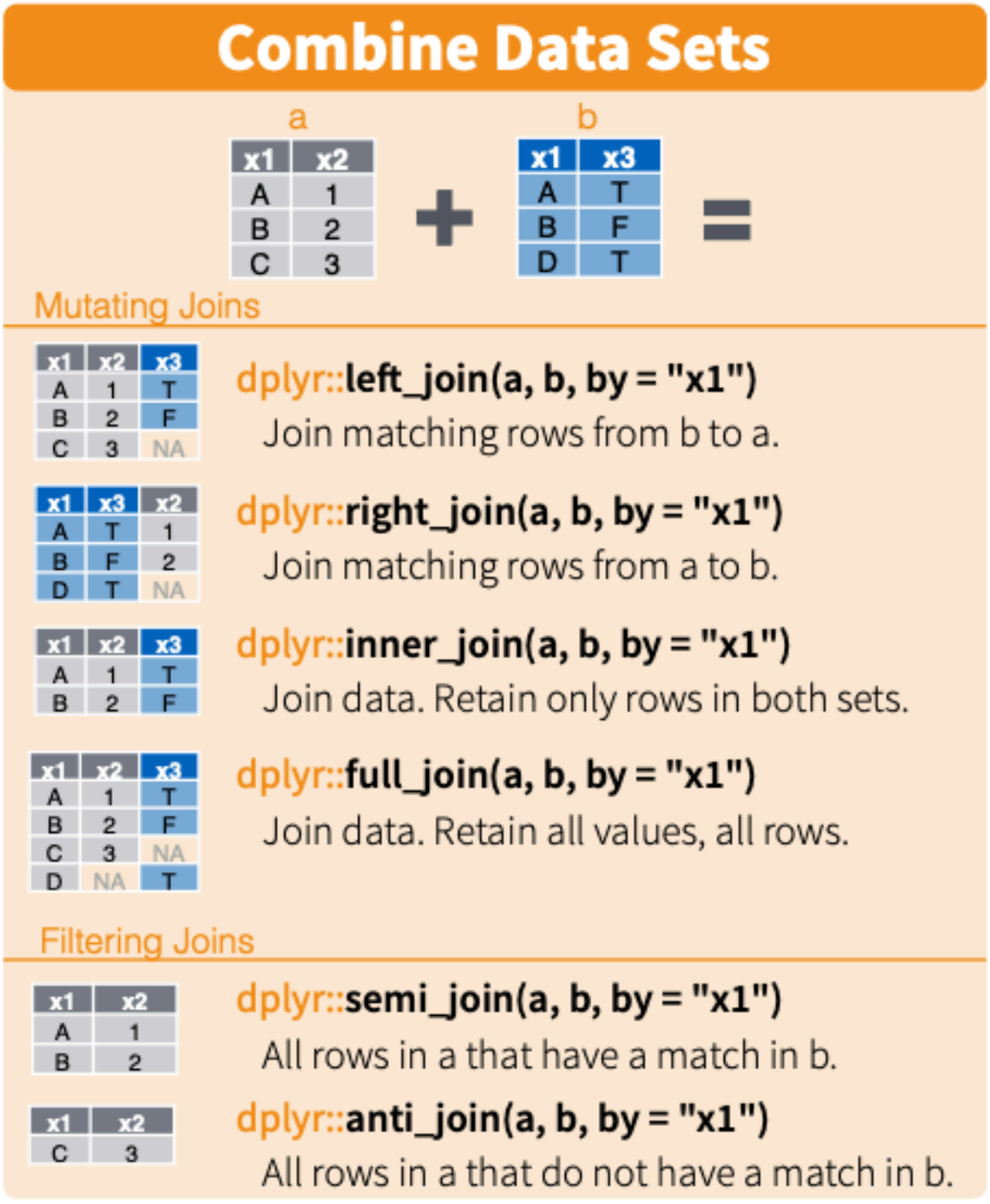
Another package that can give a brief overview of your data is `summarytools`

```
library(summarytools)
view(dfSummary(df)) # use view lowercase to see html output in the Rstudio viewer pane
```

3.2.4 Joining data sets

Sometimes the data you need lives in two tables. `dplyr` from the `tidyverse` package makes it easy to join your data sets together. In order to join two tables together, you'll need a shared unique identifier across the two tables.

Below are all the ways you can join two data sets using R and the corresponding `dplyr` functions.



You can view this image and additional ways to transform data sets on the RStudio Data Wrangling Cheat Sheet.

In Appendix C, I've generated a fake dataset with corresponding ResponseId's that match to the survey data set (df).

Below I use a left join to merge respondent data table with the survey data table.

```r
df <- df %>% left_join(respondent_data, by = c("ResponseId" = "ResponseId"))

# View merged data sets
skim(df)
```

```
## Skim summary statistics
##  n obs: 502
##  n variables: 36
##
## -- Variable type:character ---------------------------------------------------
##             variable missing complete   n min max empty n_unique
##                  age       0      502 502   3   8     0        4
##  DistributionChannel       0      502 502   4   4     0        1
##                email       0      502 502   9  23     0      405
##           first_name       0      502 502   2  11     0      373
##               gender       0      502 502   4  17     0        4
##                  job       0      502 502   4  59     0      346
##                 name       0      502 502   7  27     0      502
##         phone_number       0      502 502  11  20     0      502
##                   Q1       0      502 502   5  14     0        6
##                   Q2       0      502 502  18  34     0        5
##               Q3_10     184      318 502   5   5     0        1
##           Q3_10_TEXT     184      318 502  51 135     0      318
##                Q3_4     201      301 502  26  26     0        1
##                Q3_5     165      337 502  22  22     0        1
##                Q3_6     174      328 502  21  21     0        1
##                Q3_7     172      330 502  19  19     0        1
##                Q3_8     184      318 502  18  18     0        1
##                Q3_9     162      340 502  23  23     0        1
##                  Q4       0      502 502  11  22     0        7
##                  Q5       0      502 502  53 134     0      502
##           ResponseId       0      502 502  17  17     0      502
##               Status       0      502 502  11  11     0        1
##
## -- Variable type:logical ----------------------------------------------------
##            variable missing complete   n mean           count
##   ExternalReference     502        0 502  NaN             502
##            Finished       0      502 502    1 TRU: 502, NA: 0
##           IPAddress     502        0 502  NaN             502
##      RecipientEmail     502        0 502  NaN             502
##  RecipientFirstName     502        0 502  NaN             502
##   RecipientLastName     502        0 502  NaN             502
##        UserLanguage     502        0 502  NaN             502
##
## -- Variable type:numeric ----------------------------------------------------
##               variable missing complete   n     mean   sd      p0     p25
##  Duration (in seconds)       0      502 502    0.024 0.15    0       0
##       LocationLatitude       0      502 502   37.77  0      37.77   37.77
##      LocationLongitude       0      502 502 -122.41  0    -122.41 -122.41
##               Progress       0      502 502  100     0     100     100
##      p50     p75    p100     hist
##     0       0       1
##    37.77   37.77   37.77
##  -122.41 -122.41 -122.41
```

```
##   100     100     100
##
## -- Variable type:POSIXct -------------------------------------------------
##      variable missing complete   n        min        max     median
##       EndDate       0      502 502 2019-01-15 2019-01-15 2019-01-15
##  RecordedDate       0      502 502 2019-01-15 2019-01-15 2019-01-15
##     StartDate       0      502 502 2019-01-15 2019-01-15 2019-01-15
##  n_unique
##        74
##        74
##        74
```

3.2.5 Removing duplicate values

Respondents may come back to the survey, or try to take the survey a second time on a new device. To ensure a respondent isn't counted more than once in a survey, be sure to check for duplicate values by using a unique identifier. Common unique indentifiers include: email, embedded user id, or IP address.

View duplicates using janitor package

```
library(janitor)

df %>% get_dupes(email) # get_dupes is a function available through janitor, can use more than one colu
```

```
## # A tibble: 122 x 37
##    email dupe_count StartDate           EndDate             Status
##    <S3:>      <int> <dttm>              <dttm>              <chr>
##  1 dr@g~         13 2019-01-15 13:36:53 2019-01-15 13:36:53 Surve~
##  2 dr@g~         13 2019-01-15 13:36:59 2019-01-15 13:36:59 Surve~
##  3 dr@g~         13 2019-01-15 13:37:01 2019-01-15 13:37:01 Surve~
##  4 dr@g~         13 2019-01-15 13:37:09 2019-01-15 13:37:09 Surve~
##  5 dr@g~         13 2019-01-15 13:37:21 2019-01-15 13:37:21 Surve~
##  6 dr@g~         13 2019-01-15 13:37:21 2019-01-15 13:37:21 Surve~
##  7 dr@g~         13 2019-01-15 13:37:23 2019-01-15 13:37:23 Surve~
##  8 dr@g~         13 2019-01-15 13:37:24 2019-01-15 13:37:24 Surve~
##  9 dr@g~         13 2019-01-15 13:37:25 2019-01-15 13:37:25 Surve~
## 10 dr@g~         13 2019-01-15 13:37:27 2019-01-15 13:37:27 Surve~
## # ... with 112 more rows, and 32 more variables: IPAddress <lgl>,
## #   Progress <dbl>, `Duration (in seconds)` <dbl>, Finished <lgl>,
## #   RecordedDate <dttm>, ResponseId <chr>, RecipientLastName <lgl>,
## #   RecipientFirstName <lgl>, RecipientEmail <lgl>,
## #   ExternalReference <lgl>, LocationLatitude <dbl>,
## #   LocationLongitude <dbl>, DistributionChannel <chr>,
## #   UserLanguage <lgl>, Q1 <chr>, Q2 <chr>, Q3_4 <chr>, Q3_5 <chr>,
## #   Q3_6 <chr>, Q3_7 <chr>, Q3_8 <chr>, Q3_9 <chr>, Q3_10 <chr>,
## #   Q3_10_TEXT <chr>, Q4 <chr>, Q5 <chr>, name <chr>, first_name <chr>,
## #   job <chr>, phone_number <chr>, gender <chr>, age <chr>
```

Manual way to view duplicates

```
u_id <- quo(email) # Store unique identifier column, can be IP address, email, etc.

df %>% group_by(!!u_id) %>%
       tally() %>%
       filter(n > 1)
```

```
## # A tibble: 25 x 2
##    email                 n
##    <S3: glue>        <int>
##  1 dr@gmail.com         13
##  2 dr@hotmail.com        6
##  3 dr@me.com            11
##  4 dr@outlook.com       12
##  5 dr@yahoo.com         11
##  6 hope@yahoo.com        2
##  7 lauren@outlook.com    2
##  8 miss@me.com           5
##  9 miss@outlook.com      3
## 10 miss@yahoo.com        2
## # ... with 15 more rows
```

You'll want to remove duplicate responses, and keep the most recent response.

```
library(lubridate) # load library for converting datetimes

#Remove duplicate emails, keep most recent submission
df <- df %>%
  mutate(EndDate = as_datetime(EndDate, tz = "America/Los_Angeles")) %>% # converts column to a datetim
  filter(!is.na(!!u_id)) %>%
  group_by(!!u_id) %>%
  slice(which.max(EndDate)) %>%
  ungroup()
```

Appendix A

Setting up R

A.1 Package installation

You'll want to install the following packages:

```
library(tidyverse)
```

Appendix B

Setting up python

```
# Pandas makes working with data tables easier
import pandas as pd

# Numpy is a library for working with Arrays
import numpy as np

# Module for plotting graphs
import matplotlib.pyplot as plt
import seaborn as sns

# SciPy implements many different numerical algorithms
import scipy.stats as stats
import collections
```

Appendix C

Generating fake data

Here's the code I used to create the respondent information table

```
library(charlatan) # library of fake data
library(glue) # library for pasting together variables

email_domains <- c("@gmail.com", "@hotmail.com", "@outlook.com", "@me.com", "@yahoo.com")

respondent_data <- ch_generate('name', 'job', 'phone_number', n = nrow(df)) %>%
                    separate(name, "first_name", extra = "drop", remove=FALSE) %>%
                    mutate(email = glue("{first_lower}{email_domain}",
                                        first_lower = tolower(first_name),
                                        email_domain = sample(email_domains, nrow(df), replace=TRUE)
                           gender = sample(c("male", "female", "other", "prefer not to say"), nrow(df),
                           age = sample(c("Under 18", "18-34", "35-54", "55+"), nrow(df), replace=TRUE,
       )

# add ResponseId column from survey sample
respondent_data <- df %>% select(ResponseId) %>%
  bind_cols(respondent_data)

write_csv(respondent_data, "./sample_data/respondent_data.csv") # Store data in sample_data folder
```

```
skim(respondent_data)
```

```
## Skim summary statistics
##  n obs: 502
##  n variables: 8
##
## -- Variable type:character ------------------------------------------------
##      variable missing complete   n min max empty n_unique
##           age       0      502 502   3   8     0        4
##         email       0      502 502   9  23     0      405
##    first_name       0      502 502   2  11     0      373
##        gender       0      502 502   4  17     0        4
##           job       0      502 502   4  59     0      346
##          name       0      502 502   7  27     0      502
##  phone_number       0      502 502  11  20     0      502
##    ResponseId       0      502 502  17  17     0      502
```

Bibliography

Ben Marwick, Carl Boettiger, L. M. (2018). Packaging data analytical work reproducibly using r (and friends). *PeerJ*.

Rohrer, C. (2014). When to use which user-experience research methods. *Nielsen Norman Group*.